View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by S

Descubrimiento e interacción de servicios móviles ubicuos utilizando Bluetooth y WiFi

Ricardo Andrés Fajardo
rafajardo@unicauca.edu.co

Víctor Fabián Miramá
vmirama@unicauca.edu.co

Oscar Mauricio Caicedo R.
omcaicedo@unicauca.edu.co

Javier Mesa Durango
jmesa@unicauca.edu.co

Francisco Orlando Martínez P.
fomarti@unicauca.edu.co
Grupo de Ingeniería Telemática, Universidad del Cauca, Popayán, Cauca, Colombia

Fecha de recepción: 19-10-2007 Fecha de seleccción: 18-04-2008 Fecha de aceptación: 15-01-2008

ABSTRACT

Wireless Communications, especially cellular systems have an important role in actual communication systems. Mobility is an intrinsic characteristic for people that motivate the creation of and commerce. This architecture pretends to adapt some existent technologies for electronic billing and payment to the Colombian context.

KEY WORDS

Ubiquitous computing, RFID, NFC, Certified entity, Cryptography.

RESUMEN

Las comunicaciones inalámbricas y en especial la telefonía celular, desempeñan un rol importante en las comunicaciones actuales, la movilidad es una característica intrínseca de las personas, que origina la necesidad de crear servicios que se ajusten a los deseos de los clientes. Es así como ha surgido la computación ubicua, haciendo referencia a la presencia e interacción, casi imperceptibles de la computación para la prestación de servicios en cualquier instante.

Sin embargo, el ofrecimiento de servicios ubicuos presenta un problema y es que no existe un protocolo que cumpla con los requerimientos de identificación de contexto, movilidad y flexibilidad que estos demandan. Este artículo describe la construcción de un piloto de servicios ubicuos en un ambiente móvil, creado como herramienta de pruebas para el proyecto, Protocolo de Descubrimiento e Interacción de Servicios Ubicuos en un Ambiente Móvil, con el objetivo de aportar e impulsar la investigación y el desarrollo de este tipo de servicios, respaldado por el grupo de interés en Desarrollo de Aplicaciones Inalámbricas y/o para Dispositivos Móviles (W@pColombia) del Grupo de Ingeniería Telemática (GIT) de la Universidad del Cauca.

PALABRAS CLAVE

Bluetooth, Descubrimiento de servicios, Movilidad, Servicios Ubicuos, Servicios Web, WiFi.

Clasificación Colciencias: Tipo 1

1. INTRODUCCIÓN

El continuo crecimiento de internet y las comunicaciones inalámbricas transforman la sociedad de la información en una sociedad más ágil y exigente, cuyos requerimientos tecnológicos crecen al ritmo de las tecnologías. En este momento es difícil pensar en vivir sin un teléfono celular, dispositivo móvil o acceso a internet. Las personas día a día buscan facilitar, agilizar y organizar sus labores con dispositivos tan comunes como los dispositivos móviles.[1]

La tendencia hacia el uso de dispositivos móviles, motiva el desarrollo de servicios adecuados a las necesidades de los clientes, de tal manera que un usuario tenga la posibilidad de escoger un servicio y adecuarlo a sus necesidades y además tener acceso a él en cualquier instante, sitio y dispositivo. Otra característica de los servicios ubicuos es la disminución de la interacción entre el usuario y el dispositivo, es decir, los usuarios desean que los servicios sean más intuitivos y que funcionen autónoma y automáticamente teniendo en cuenta sus preferencias.[2]

En la búsqueda del desarrollo de estos servicios, se han realizado diferentes proyectos, como MoBe de la universidad de Udine en Italia,[3] que definió una arquitectura modular, en la cual existen servidores que envían aplicaciones software a los dispositivos móviles, continuamente, mediante el proceso de push,[4] según el contexto del usuario; el proyecto PUMAS, el cual desarrolló un framework[5] para ayudar al usuario final, en el manejo de aplicaciones utilizadas por las personas en sus dispositivos móviles[6] y el piloto de servicio ubicuo creado por los laboratorios NTT.[7]

En los proyectos mencionados, el ofrecimiento de servicios ubicuos tiene diferentes inconvenientes, tales como los dispositivos utilizados o la compatibilidad de los mismos, lo que interrumpe la estandarización de los servicios ubicuos, además, en las investigaciones actuales no existe un protocolo con características de identificación de contexto, si se tienen en cuenta las necesidades del usuario, la movilidad y flexibilidad de éste, en otras palabras, que posibilite el ofrecimiento de estos servicios. Debido a esto, el grupo de ingeniería telemática de la Universidad del Cauca desarrolla proyectos paralelos de definición funcional y arquitectónica de protocolos para servicios ubicuos, su validación e implementación. En uno de ellos se pretende crear un protocolo de ofrecimiento e interacción de servicios ubicuos, que se ajuste a los requerimientos de ubicuidad, que sea validado mediante el piloto descrito en este documento y que aporte una alternativa eficiente que mejore su ofrecimiento.

La construcción del piloto se desarrolló conforme a los requerimientos del ofrecimiento de un servicio ubicuo, como son la reducción de la interacción con el usuario y los servicios basados en contexto; proceso que se describe en las secciones: II, que define los conceptos relacionados; III, donde se describe la arquitectura básica del piloto, su construcción, las pruebas realizadas y los resultados obtenidos; y en IV, se plantean las conclusiones y los trabajos futuros.

2. SERVICIOS UBICUOS

A. Definición

Fue Mark Weiser quien definió la computación ubicua o computación

invisible, como "Computación incrustada en el ambiente, disponible en todas partes para ayudar al usuario en la culminación de sus tareas".[1,2] En otros términos la computación ubicua hace referencia a la presencia e interacción, casi imperceptibles, de la computación para la prestación de servicios,[2] y en relación con los servicios es la posibilidad de tener acceso a ellos desde cualquier sitio, en cualquier tiempo y sin necesidad de interacción permanente cliente-dispositivo, de tal manera que el usuario tenga acceso al servicio sin solicitarlo directamente, basta con conocerle sus características para identificar sus preferencias y ofrecerle un servicio a su medida y en cualquier parte con mayor libertad.[8]

En comparación con los servicios tradicionales, los servicios ubicuos se caracterizan por estar basados en condiciones físicas, como el contexto, el clima, el estado de ánimo y la flexibilidad, es decir que se pueden acceder desde cualquier dispositivo, facilitando las tareas de las personas.[2]

B. Protocolos de descubrimiento e interacción de servicios

Para la construcción de servicios ubicuos es necesario implementar protocolos que den soporte al descubrimiento y la interacción de servicios. Existen en la actualidad diferentes soluciones como Jini, SLP (Service Location Protocol), UPnP (Universal Plug and Play), SDP (Service Discovery Protocol) o Salutation, encargados de estas tareas en las redes cableadas e inalámbricas.

De manera general estos protocolos definen los principales actores de un sistema, identifican roles como usuario, proveedor y el motor o servidor de descubrimiento de servicios, que es el elemento contenedor de la información de los servicios disponibles para los usuarios del sistema y se encarga de facilitar la comunicación entre usuario y proveedor. Además, definen las políticas de comunicación y negociación para la disponibilidad y el uso de los servicios.[9-13]

Sin embargo, estos protocolos no han sido diseñados para ofrecer servicios ubicuos, debido a que las redes en las que se implementan son estáticas, cuyos parámetros y características son definidos antes de su instalación, además el número de dispositivos o clientes, se mantiene constante y su contexto no cambia. Sin embargo, las características y roles mencionados son una base importante para la construcción de soluciones que posibiliten el ofrecimiento de este tipo de servicios y se utilizaron en el trabajo presentado en este artículo.

C. Protocolos de transporte y acceso

Dentro de los protocolos de transporte inalámbricos más difundidos se tomó como referencia a bluetooth y WiFi (Wireless Fidelity), debido a su gran importancia ya que proporcionan la movilidad y flexibilidad necesarias para la construcción del piloto de servicios ubicuos.

Bluetooth:[14] está definido en el estándar 802.15.1, es utilizado para facilitar la comunicación entre los dispositivos móviles, el servidor y el proveedor de servicios, su importancia radica en el descubrimiento de servicios y la posibilidad de interactuar sin intervención administrativa, con otros dispositivos; aporta al

proyecto el soporte a la movilidad del medio y minimiza la interacción cliente – dispositivo. Con bluetooth se agrega un grado de confiabilidad al sistema, debido a sus tres niveles de seguridad que pueden ser utilizados para proporcionar mayor confianza en el servicio.

WiFi:[15] definido en el estándar 802.11, su utilización en el proyecto se debe a la libertad que proporciona al usuario, gracias a que el rango de cobertura WiFi es mayor que el de bluetooth, extendiendo el ofrecimiento de los servicios, obteniendo de esta manera un entorno ubicuo más eficiente. Por otro lado, al ser WiFi una tecnología o un estándar que empieza su ingreso en el mercado de los dispositivos móviles celulares, dota al proyecto de un amplio margen de permanencia y aceptación.

NFC:[16,17] tecnología definida en el estándar ISO/IEC 18092 de 2004, es utilizada para realizar el pago de servicios y evitar el uso de dinero en efectivo y los métodos tradicionales de pago, agilizando la vida del usuario.

RFID:[18] especificada en los estándares ISO/IEC 15961 y 15962 de 2004, es empleada para la identificación y ejecución de procesos automáticos. En este proyecto es el soporte del registro y pago de servicios al utilizar la información financiera del usuario para agilizar las transacciones requeridas.

Cabe anotar que si bien estas tecnologías no proporcionan mecanismos complejos de seguridad, el limitado rango de cobertura de la señal de radio frecuencia, hace que sea difícil interceptar una comunicación utilizando NFC o RFID. El Grupo de Ingeniería Telemática está adelantando un proyecto para pagos de proximidad seguros, que proporcionen mayor confianza en este tipo de servicios de pago, para complementar el presente trabajo.

3. CASOS DE ESTUDIO

A. Ciclo de vida del servicio ubicuo

Teniendo en cuenta investigaciones previas,[9-13] se ha definido el ciclo de vida de un servicio ubicuo (Figura 1). Se toma como referencia el entorno de un centro comercial, en el cual se ofrece al usuario el servicio de promoción y venta de productos de acuerdo con su perfil.

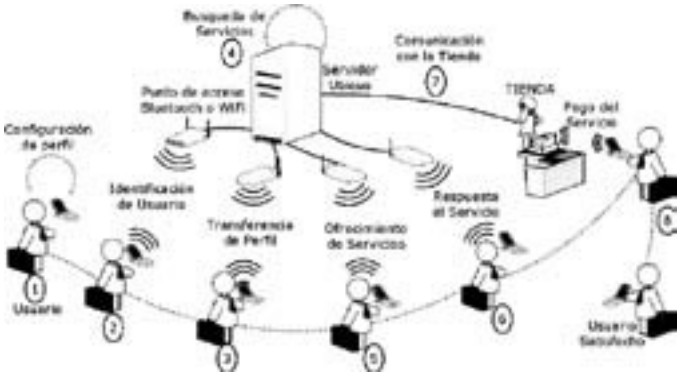

Figura 1. Ciclo de vida

Para acceder al servicio el usuario debe configurar su perfil, especificando la línea de productos o servicios sobre los que desea recibir información[1] y ponerse visible para el sistema, para poder ser identificado.[2] Cuando el servidor identifica al usuario, realiza una solicitud de información de perfil,[3] después de recibir dicha información, compara las preferencias del usuario con los servicios que están disponibles[4] con el fin de seleccionar aquellos que se ajusten más a las mismas. El sistema informa al usuario de los servicios disponibles y éste selecciona, si lo desea, alguno de ellos. Si el usuario decide aceptar al menos un servicio, envía la confirmación[6] y el sistema a su vez informa a la tienda o almacén la transacción realizada,[7] la cual se registra en la etiqueta RFID del dispositivo móvil del usuario, para que éste se acerque a la tienda y la complete, recoja el producto comprado y descuente de su cuenta bancaria el costo correspondiente.

Con el ciclo de vida definido se decidió desarrollar dos pilotos que sirvan como base para la construcción del protocolo de descubrimiento e interacción de servicios ubicuos, aportando las características que faciliten su utilización en cualquier dispositivo y protocolo de transporte.

B. Piloto de servicios ubicuos sobre bluetooth

Arquitectura

Para la construcción del piloto bluetooth se diseñó la arquitectura de la Figura 2, que muestra los módulos funcionales de las entidades principales del piloto.

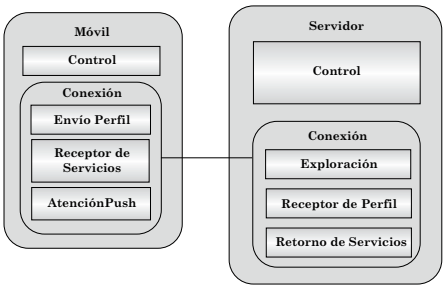

Figura 2. Arquitectura del piloto bluetooth

Servidor

Control: es el encargado de controlar los procesos necesarios para el descubrimiento e interacción de servicios ubicuos, los cuales son: Exploración del entorno para encontrar dispositivos móviles, comparación de los servicios con el perfil del usuario, creación de la respuesta con los servicios encontrados y manejo de los mensajes push.

Conexión: es el encargado de crear y establecer conexiones con cada dispositivo móvil encontrado, distribuyendo su trabajo en tres sub - módulos. El primero, Exploración, busca continuamente nuevos dispositivos para iniciar el descubrimiento de servicios, el segundo, Recepción de perfil, se encarga de intercambiar el perfil del usuario desde el móvil hacia el servidor y el tercero, Retorno de servicios, envía los servicios que fueron encontrados al dispositivo móvil.

Móvil

Control: es el encargado de manejar las interfaces para configurar el perfil del usuario, además de controlar los procesos para que el servidor realice el descubrimiento y la interacción de servicios.

Conexión: por medio de este módulo se lleva a cabo el proceso de descu-

brimiento e interacción de servicios ubicuos, a través de los siguientes sub - módulos:

Envío de perfil: se encarga de establecer la conexión bluetooth con el servidor para intercambiar el perfil del usuario.

Receptor de servicios: tiene la función de establecer una conexión con el servidor para que éste envíe los servicios descubiertos.

Atención Push: permite autoiniciar la aplicación, cuando llegan los push desde el servidor, para realizar el envío del perfil o recepción de servicios.

Diagrama de paquetes
El piloto bluetooth está implementado con el lenguaje de programación java en sus versiones estándar y micro, para el servidor ubicuo y la aplicación móvil, respectivamente; además se utilizó el patrón de diseño Modelo Vista Controlador – MVC, el cual ayuda a organizar las clases en paquetes de acuerdo con la función que cumplen. La Figura 3 muestra el diagrama de paquetes de la aplicación móvil. Vista es el paquete que contiene los formularios y listas que permiten la interacción del usuario con la aplicación, ya que a través de estos se pueden ver las preferencias que el usuario ha creado, los servicios que el servidor encuentra y habilitar a la aplicación para el envío de las preferencias al servidor.

Las clases más importantes del lado de la aplicación cliente se describen a continuación:

control.SUMidlet: hereda de la clase MIDlet y se encarga de inicializar todos los objetos y además realizar

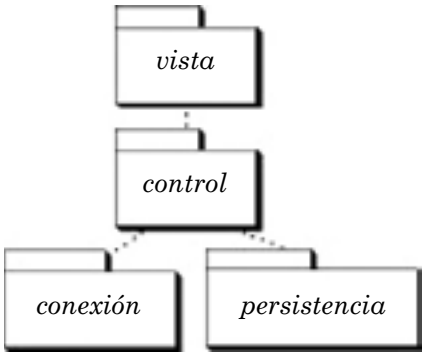

Figura 3. Diagrama de paquetes del móvil

el registro del Push Bluetooth, para que la aplicación pueda auto iniciarse cuando reciba una solicitud de conexión bluetooth.

conexion.ConnectionManager: se encarga de establecer y controlar las conexiones bluetooth con el servidor, para intercambiar la información correspondiente a las preferencias y los datos de cuenta del usuario.

persistencia.RMSManager: maneja un RecordStore,[19] el cual almacena todas las preferencias del usuario, a través de sus métodos que permiten agregar, editar y eliminar preferencias. También proporciona el almacenamiento local de los datos bancarios del usuario.

La Figura 4 muestra el diagrama de paquetes de la aplicación del servidor, el cual cuenta con dos paquetes conexión y control. El paquete bluecove-JSR82, permite que se utilice el API de bluetooth desde una aplicación java convencional.

Figura 4. Diagrama de paquetes del servidor

Las principales clases del lado de la aplicación servidora son las siguientes:

control.Manager: encargada de llevar el control de todo proceso, primero hace una exploración del entorno para buscar usuarios, luego recibe sus preferencias para hacer la comparación, posteriormente, crea el mensaje para enviarlo al usuario y finalmente envía un Push Bluetooth para que la aplicación en el dispositivo móvil se autoejecute y muestre los servicios encontrados.

conexion.GetProfile: abre una conexión bluetooth y busca usuarios que desean intercambiar sus preferencias. La búsqueda de usuario es continua y se crea una sesión para cada uno.

conexion.SendResponse: construye el mensaje con el resultado de la comparación de las preferencias del usuario, de acuerdo con los servicios que existen, para enviarlo al dispositivo móvil.

conexion.Push: es la encargada de enviar el Push Bluetooth y transmitir el resultado de la comparación de los servicios, para que la aplicación los despliegue en el dispositivo móvil, mediante un formulario.

Descripción del Servicio (ver Figura 5). Este piloto está conformado por un servidor que maneja un dispositivo Bluetooth USB, y por un móvil que tiene empotrada la tecnología bluetooth. Una vez el servidor está en marcha utiliza el sub - módulo de Exploración para buscar dispositivos móviles dentro del área de cobertura bluetooth, informando al módulo de control cada vez que encuentra uno. Una vez hallado un dispositivo móvil, el módulo Control solicita el perfil del usuario enviando un mensaje Push Bluetooth a través del sub - módulo Receptor de Perfil; cuando el móvil recibe el push, el AMS (Application Management System)[4] detecta la conexión entrante y activa el midlet para que el sub - módulo Atención Push analice el mensaje y determine qué proceso se debe realizar e informar al módulo Control. Como el push es de solicitud de perfil, el módulo Control lee el perfil almacenado del usuario y lo envía al servidor a través del sub - modulo Envío de Perfil, mediante una conexión bluetooth previamente establecida entre el móvil y el servidor. Cuando el sub - módulo Receptor de Perfil del servidor ha recibido el perfil, envía éste al Control para realizar el proceso de descubrimiento de servicios. Luego de encontrar los servicios adecuados para el perfil del usuario, el Control envía nuevamente un push Bluetooth; pero esta vez indica los servicios encontrados para de esta manera empezar la interacción de servicios. Cuando el dispositivo móvil recibe el push y determina que tiene servicios disponible para él, a través del sub - módulo Atención Push, informa al Control para que abra una conexión bluetooth mediante el sub - módulo Receptor de Servicios, el cual recibe los servicios que han sido descubiertos para posteriormente entregarlos al Control, que los desplegará por medio de interfaces gráficas al usuario.

Resultados

La experimentación del piloto se ejecutó utilizando un servidor, con sistema operativo Windows XP Versión 2002 Service Pack 2, Procesador Pentium(R) 2.80GHz y memoria

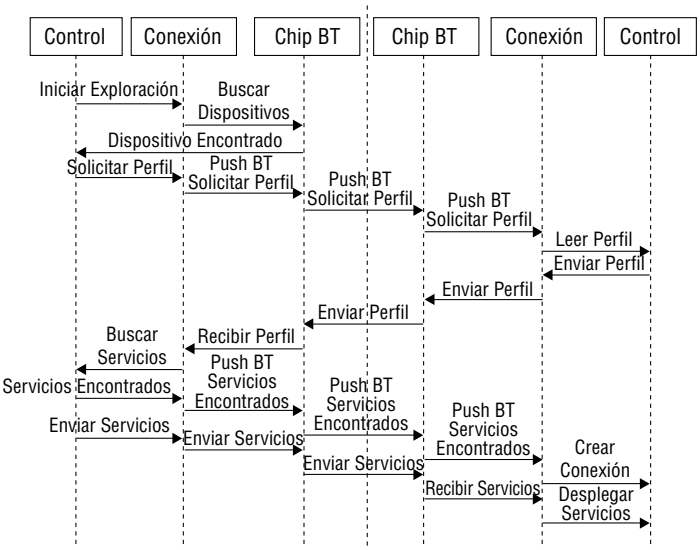

Figura 5. Diagrama de secuencia Bluetooth

RAM de 1GB, los emuladores de Nokia Carbide.j 1.5 y WTK 2.5. Para el ambiente real el dispositivo móvil utilizado fue el Nokia N90 (Figura 6).

Las pruebas y los resultados obtenidos se relacionan a continuación.

Consumo de Memoria: La Figura 7 muestra los valores de consumo de memoria durante los procesos fundamentales del servicio, las medidas se realizaron utilizando el monitor de memoria del WTK y para ello se tomó el promedio de cien medidas.

El consumo de memoria tiene valores que disminuyen o aumentan, lo cual depende de la importancia del proceso ejecutado. En este sentido se puede notar que el consumo más bajo es para el proceso de descubrimiento e interacción de servicios, característica importante para el soporte de la movilidad del usuario.

A pesar de que los otros valores de memoria consumida son altos, comparados con otro tipo de aplicaciones están por debajo del valor mínimo (4Mb a 8Mb para teléfonos básicos y

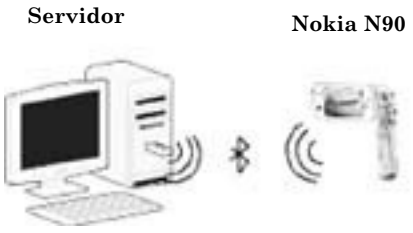

Figura 6. Entorno de pruebas

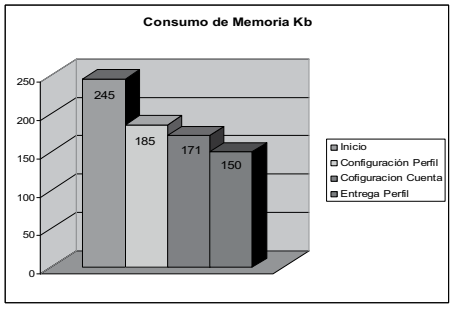

Figura 7. Consumo de memoria

16Mb a 128Mb para teléfonos gama alta,[20]) en la práctica no representan un inconveniente para la prestación de servicios ubicuos en dispositivos móviles ya que estos se orientan a los nuevos terminales que típicamente tienen altas prestaciones en memoria y procesamiento.

Tiempos de Respuesta: Las Figuras 8 y 9 muestran los tiempos de ejecución de los procesos necesarios para tener acceso al servicio, algunos de los cuales dependen directamente del usuario y otros del servicio. Estos tiempos se midieron utilizando el método *System.currentTimeMillis()* ejecutado dentro de la aplicación, al inicio y al final de cada proceso, tomando el promedio de cien medidas.

En cuanto al tiempo de respuesta se puede notar una diferencia entre la simulación y la aplicación real, que se debe al ambiente de simulación aislado, que no se ve afectado por algún retardo en transmisión, como es el caso del ambiente real, además las respuestas de un computador son más rápidas por la mayor capacidad de procesamiento que tiene, frente a la de un dispositivo celular. También cabe anotar que, por el hecho que la aplicación móvil no está certificada, cada vez que ésta realiza una conexión o se va a autoiniciar pide la validación del usuario, razón por la cual el tiempo de respuesta se incrementa. Por esto, para una aplicación comercial es necesario tener un certificado digital que valide a la aplicación como segura y la interacción dispositivo/cliente disminuya y sea transparente para el usuario.

Tamaño de los mensajes: En la Figura 10 se relaciona el tamaño de los mensajes enviados durante la configuración y acceso a los servicios, medido utilizando el monitor de red del WTK. Esta prueba se realizó para el caso de envío del perfil, para la preferencia Categoría - Deporte, Tipo - Fútbol, Artículo - Camiseta. Para un perfil con más preferencias, el tamaño del mensaje tendrá un valor múltiplo, debido a que este es el tamaño base para las preferencias.

Se debe tener en cuenta que para este piloto el perfil no comprende la información personal del usuario, por lo cual el tamaño de los mensajes es pequeño.

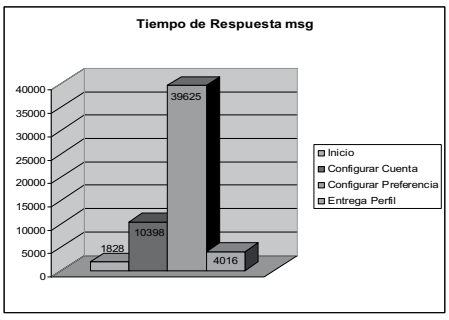

Figura 8. Tiempos de respuesta en emulación

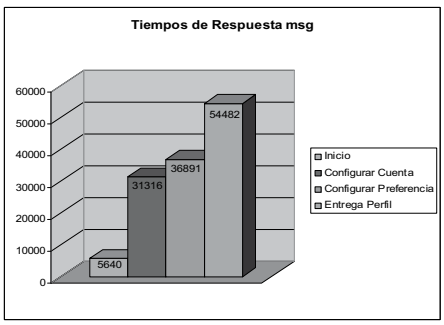

Figura 9. Tiempos de respuesta reales

C. Piloto de servicios ubicuos sobre WiFi

Arquitectura

La arquitectura diseñada para el piloto WiFi se muestra en la Figura 11, donde se puede observar los módulos descritos a continuación.

Servidor

Control: además de registrar los usuarios descubiertos y los servicios, este módulo realiza las mismas funciones que el módulo Control del piloto bluetooth.

Conexión: es el encargado de crear y establecer conexiones WiFi con cada dispositivo móvil encontrado, distribuyendo su trabajo en tres sub - módulos. El primero, Exploración, busca continuamente nuevos dispositivos en la red WiFi para posteriormente iniciar el descubrimiento de servicios, el segundo, TransReceptor, se encarga de intercambiar el perfil del usuario desde el móvil hacia el servidor y también de regresar los servicios encontrados. Y el tercero, Notificación, anuncia los móviles encontrados, a través de mensajes Push, que han sido descubiertos con miras a que se preparen para iniciar el proceso de descubrimiento, y también advierte cuando hay servicios descubiertos para el dispositivo móvil y así proceder a iniciar la interacción.

Registro: tiene la función de hacer el registro, en una base de datos, de los dispositivos móviles encontrados y de los servicios existentes para cada uno.

Móvil

Control: realiza las mismas funciones que el módulo Control del piloto Bluetooth.

Conexión: a través de este módulo se hacen las conexiones que permiten el envío del perfil y la recepción de los servicios. También es el encargado de recibir las notificaciones del usuario las cuales señalan el proceso a desarrollar, tareas que están a cargo de los sub - módulos TransReceptor y Notificación, respectivamente.

Diagrama de paquetes

El piloto se implementó en java en sus versiones estándar y micro para el servidor ubicuo y la aplicación móvil, respectivamente. En esta última se utilizó el patrón de diseño Modelo Vista Controlador – MVC, el cual ayudó a organizar las clases en paquetes, de acuerdo con la función que cumplen.

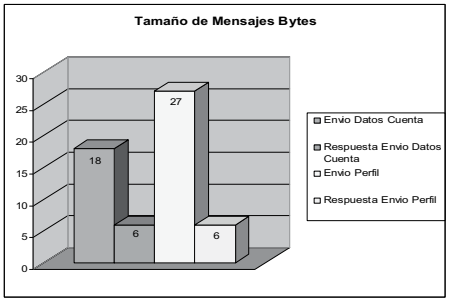

Figura 10. Tamaño de mensajes bytes

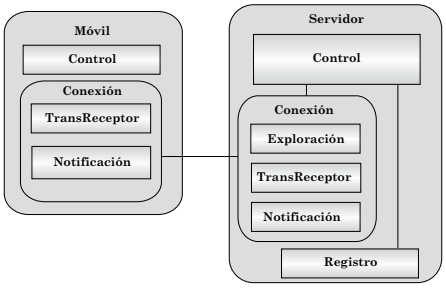

Figura 11. Arquitectura del piloto WiFi

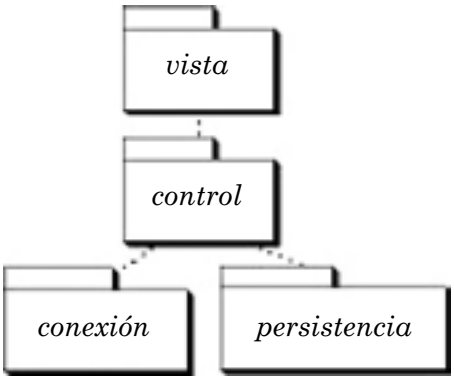

Figura 12. Diagrama de paquetes del móvil

La Figura 12 muestra el diagrama de paquetes de la aplicación móvil, del cual las clases más importantes son:

control.ManagerPreference: es la encargada de agregar, editar y eliminar las preferencias, así como también de actualizar los datos personales del usuario, además controla la navegación entre las diferentes interfaces.

control.ManagerServices: se encarga de construir los formularios que despliegan los servicios encontrados por el servidor, y permite la navegación entre ellos.

control.Connector: se encarga de manejar la conexión con el servidor y hacer peticiones.

persistencia.RMSManager: esta clase se encarga de manejar un *RecordStore*, el cual almacena todas las preferencias del usuario y su perfil, a través de sus métodos que permiten agregar, editar, eliminar preferencias y actualizar perfil. También almacena el identificador de sesión para recuperar los servicios que el servidor haya encontrado.

Conexion.SumoSEI_Stub: esta es una clase *Stub*, generada con la herramienta de consumo de Servicios Web del WTK, así como también todas las clases que dependen de ella, y es la que genera los mensajes SOAP (Simple Object Acces Protocol) para hacer las peticiones al servidor.

La aplicación del servidor es un Servicio Web, que atiende las peticiones de los usuarios para buscar servicios que los usuarios desean. También tiene una conexión a una base de datos (implementada en MySQL) que le permite guardar la sesión de los usuarios. La Figura 13 presenta el diagrama de paquetes de esta aplicación.

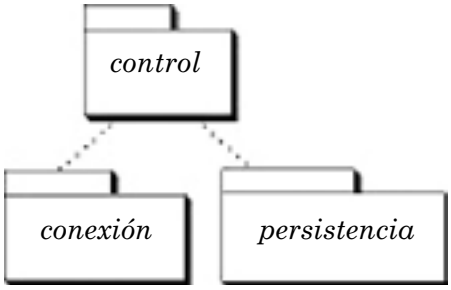

Figura 13. Diagrama de paquetes de aplicación Servidor

conexion: este paquete contiene el servicio Web con su clase de interfaz y de implementación, el cual está desplegado en el *Sun Java System Application Server* versión 9.0.

control: contiene a la clase que realiza todo el proceso de búsqueda, comparación de servicios y almacenamiento de sesión, y a la clase que genera los mensajes *Push* que se envían al Móvil.

persistencia: contiene todas las clases que permiten el acceso a la base de datos. Las mismas se generaron con la herramienta DaoGen,[21] la cual utiliza los patrones DAO y Value

Object,[22] para abstraer cada tabla como un objeto.

Descripción del Servicio

Este piloto está conformado por un servidor que está unido a un AP (Access Point) y por un móvil que tiene la tecnología WiFi dentro de sus características de comunicación. Una vez el servidor está en marcha, utiliza el sub - módulo de *Exploración* para buscar dispositivos móviles dentro del área de cobertura WiFi, informando al módulo de Control cada vez que encuentra uno. Una vez se halla un dispositivo móvil, el *Control* solicita el perfil y preferencias del usuario enviando un mensaje *Push Socket*[4] a través del sub - módulo *Notificación*; cuando el móvil recibe el push, se autoinicia la aplicación y el sub - módulo Notificación analiza el mensaje para determinar qué proceso realizar e informar al módulo de *Control*. Como el push es de solicitud de perfil el *Control* lee el perfil y las preferencias almacenadas del usuario y lo envía al servidor en el cual existe un *Servicio Web*[23] que se identifica como módulo *TransReceptor*. La forma en que se envía el perfil y preferencias del usuario desde el móvil al servidor se realiza a través del sub - módulo *TransReceptor* el cual se encarga de generar un mensaje SOAP Request y enviarlo al módulo *TransReceptor* del servidor. Cuando este módulo ha recibido el perfil y las preferencias, los envía al *Control* para efectuar el proceso de descubrimiento de servicios y además se crea un identificador de sesión el cual es enviado al móvil a través de un mensaje SOAP *Response* que el *TransReceptor* del móvil lee y transfiere al *Control* para que sea almacenado (esta información es posteriormente utilizada para solicitar servicios). Cuando el control del servidor encuentra los servicios apropiados para el perfil y preferencias del usuario, los registra en una base de datos referenciados con el identificador de sesión que se creó para el móvil. Posteriormente el Control envía nuevamente un *Push Socket*; pero esta vez indicando los servicios encontrados para así iniciar la interacción de servicios. Cuando el Móvil recibe el push y determina disponibilidad de servicios a través del sub - módulo Notificación, informa al Control para que éste lea el identificador de sesión y mediante el *TransReceptor* se cree un mensaje *SOAP* request el cual lleva el identificador para que el servidor lea de la base de datos los servicios que fueron encontrados para este móvil y que están referenciados con el identificador. Una vez se extraen de la base de datos los servicios, estos son enviados al *TransReceptor* en el móvil mediante un mensaje *SOAP response*. Finalmente cuando los servicios llegan, son encaminados al Control para su despliegue al Usuario a través de interfaces gráficas. El proceso descrito se muestra en la Figura 14.

Resultados

La experimentación del piloto se ejecutó utilizando un servidor, con sistema operativo Windows XP Versión 2002 Service Pack 2, Procesador Pentium(R) 2.80GHz y memoria RAM de 1GB, los emuladores de Nokia y Wireless Toolkit en sus versiones Carbide.j 1.5 y WTK 2.5 respectivamente. Para el ambiente real el dispositivo móvil utilizado fue el Nokia N93 y un punto de acceso a red inalámbrica Linksys (Figura 15).

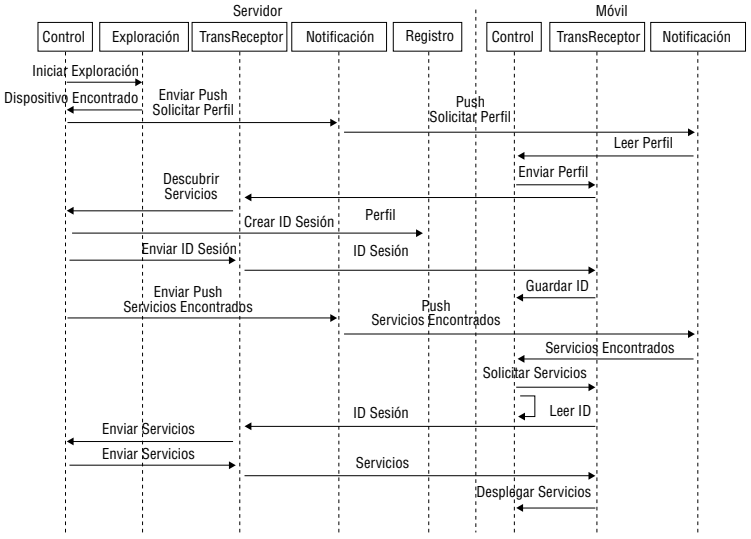

Figura 14. Diagrama de secuencia WiFi

Figura 15. Entorno de pruebas WiFi

Consumo de Memoria: en la Figura 16 se muestran los resultados obtenidos al medir el consumo de memoria durante la ejecución de la aplicación, medidas que se realizaron utilizando el monitor de memoria del WTK, tomando el promedio entre cien.

El consumo de memoria disminuye o aumenta, ello depende de la importancia del proceso a realizarse y los resultados obtenidos presentan una ocupación de memoria que no afecta el desempeño del dispositivo móvil, teniendo en cuenta que las capacidades de los dispositivos van en aumento y el tipo de terminales a los que se orientan los servicios ubicuos.

Tamaño de los mensajes: la Figura 17 corresponde a los resultados de la medida del tamaño de los mensajes que se generan para cada petición y respuesta de la aplicación móvil al servidor. Se debe tener en cuenta que la medida se hizo para la preferencia: Categoría - Deportes, Subcategoría - Artículos, Nombre - Artículos Deportivos y Prioridad 1 y para un perfil con todos sus campos llenos.

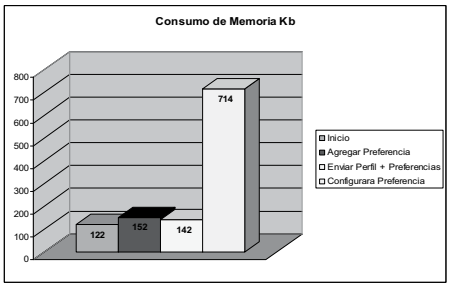

Figura 16. Consumo de memoria

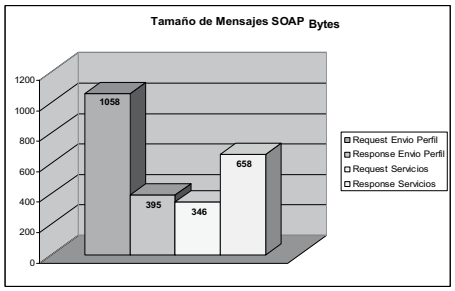

Figura 17. Tamaño de mensajes SOAP

La utilización de mensajes SOAP incrementa el consumo de ancho de banda, sin embargo, permite una mejor descripción de las preferencias del usuario, para ofrecer un mejor servicio, por lo cual para este piloto el perfil se conformó con las preferencias y los datos personales del usuario.

Tiempos de respuesta: la medida del tiempo de ejecución de los procesos importantes de la aplicación se muestra en las Figuras 18 y 19; se tuvo en cuenta los tiempos que se tardaban los procesos en emulación y en ambiente real.

En este caso, se puede observar un comportamiento similar al del piloto bluetooth. Sin embargo es importante resaltar que el tiempo empleado en editar perfil es grande debido a que contiene varios campos.

D. Comparación de los pilotos

Se considera que el piloto WiFi es más completo y estructurado por la utilización de Servicios Web, lo cual permite acercarse más a la implementación de un protocolo de descubrimiento e interacción. Es por eso que los mensajes para el piloto WiFi son más grandes que para Bluetooth, ya que toda la información intercambiada debe ir estructurada en esquemas XML para que el servidor la entienda, pero el beneficio generado es que el protocolo a desarrollar será independiente del lenguaje de programación y del protocolo de transporte, aportando así a los requerimientos de flexibilidad y escalabilidad buscados en los ambientes ubicuos.

En cuanto al consumo de memoria, para el intercambio del perfil, el resultado en ambos pilotos es muy

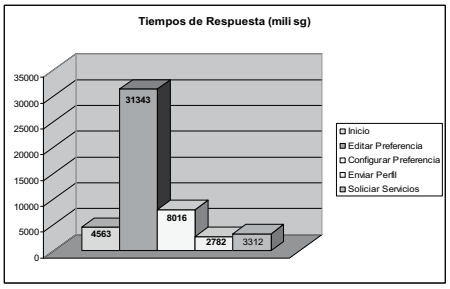

Figura 18. Tiempos de respuesta en emulación

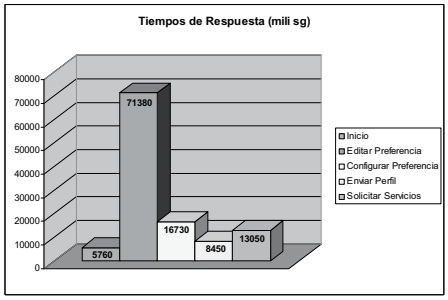

Figura 19. Tiempos de respuesta reales

similar, esto quiere decir que aunque la información intercambiada es más extensa en el de WiFi, los recursos utilizados son similares, razón por la cual se considera que el piloto WiFi es más eficiente.

En los tiempos de respuesta, el desempeño para el piloto WiFi fue superior al de Bluetooth, debido al rápido establecimiento de la conexión entre el móvil y el servidor, y a que el piloto WiFi fue probado en el dispositivo móvil Nokia N93; el cual es más rápido que el N90, utilizado en las pruebas del piloto Bluetooth.

En la Tabla 1 se muestra la comparación de los resultados obtenidos en el proceso de envío o entrega de perfil, escogida por ser una actividad que depende directamente del sistema, lo que demuestra las diferencias mencionadas en esta sección.

Tabla 1. Comparación de los pilotos

	Piloto Bluetooth	Piloto WiFi
Consumo de memoria	150Kb	142Kb
Respuesta en emulación	4016mseg.	2782mseg.
Respuesta Real	54482mseg.	8450mseg.
Tamaño de Mensaje	27Bytes	395Bytes

4. CONCLUSIONES

– Para una mayor eficiencia en la búsqueda y comparación de servicios, es necesario utilizar metadatos y generar ontologías que permitan obtener mejores resultados y al mismo tiempo alcanzar una mayor satisfacción de los usuarios.

– Cualquiera de los pilotos desarrollado es muy aplicable en el entorno comercial, en el ofrecimiento de productos y servicios, por la poca utilización de recursos y los cortos tiempos de respuesta, lo que hace a las aplicaciones atractivas para los clientes. Además, para dar soporte a una mayor cantidad de usuarios se puede implementar una red de servidores ubicuos lo cual permite una extensión de la zona de cobertura.

– El desarrollo de estos pilotos soporta el desarrollo de trabajos futuros como son la definición del protocolo de descubrimiento e interacción de servicios ubicuos, la creación de protocolos de pago de servicios ubicuos y la definición de una plataforma de servicios ubicuos. Por otro lado, se abren las puertas para la investigación acerca de métodos de búsqueda más eficientes que proporcionen soporte a los servicios basados en contexto.

5. BIBLIOGRAFÍA

1. Lee, Choonhwa and Helal, Sumi. "A Multi-tier Ubiquitous Service Discovery Protocol for Mobile Clients". Computer and Information Science and Engineering Department, University of Florida, Gainesville, 2003.
2. Weiser, Mark. "The Computer for the 21st Century". Scientific American Ubicomp Paper after Sci Am editing. 94-10, 1991.
3. Coppola, Paolo et al. "MoBe: A Framework for Context-Aware Mobile Applications". Department of Mathematics and

Computer Science, 2Department of Electrical, Management, and Mechanical Engineering, Udine, Italy, 2005.

4. Ortiz, Enrique. "The MIDP 2.0 Push Registry". Sun Microsystem. [Consulta: Agosto 2007], disponible en Web: http://developers.sun.com/mobility/midp/articles/pushreg /
5. Sun Java Studio Enterprise. "Web Application Framework Overview". Sun Microsystem. [Consulta: Septiembre 2007], disponible en Web: http://dlc.sun.com/pdf/819-0726/819-0726.pdf.
6. Carrillo, Ángela et al. "PUMAS: Un Framework que Adapta la Información en Ambientes Ubicuos". LSRIMAG Laboratory, SIGMA Team. Saint Martin d'Hères Cedex, France, 2006.
7. NTT Corporation. "Ubiquitous Services – The Technology providing ubiquitous services and evaluation with a field trial". NTT Network Services System Laboratories, 2006.
8. Comité Ejecutivo NOVOPLAY. "NOVOPLAY the Ubiquitous funny company", 2006.
9. Xin-lian, Zhou. "Service Discovery Protocol in Wireless Sensor Networks". School of Computer Science and Engineer. Hunan University of Science and Technology, 2006.
10. Bettstetter, Christian and Renner, Christoph. "A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol". Technische Universitat Munchen (TUM), Institute of Communication Networks, D-80290 Munich, Germany, 2000.
11. UPnP Forum. "UPnP Device Architecture 1.0". Versión 1.0.1, 2003. [Consulta: Septiembre 2007], disponible en Web: http://www.upnp.org/specs/arch/UPnP-DeviceArchitecture- v1.0.pdf.
12. Gryazin, Eugene A. "Service Discovery in Bluetooth". Group for Robotics and Virtual Reality. Department of Computer Science. Helsinki University of Technology, Helsinki, Finland. Published at NEC CiteSeer, Scientific Literature Digital Library. 2006.
13. Lezama, Lugo. "Modelado de dispositivos para un sistema de seguridad implementando tecnología Jini". Tesis Licenciatura, Ingeniería en Sistemas Computacionales, Capitulo 2 "Sistemas distribuidos", departamento de Ingeniería en Sistemas Computacionales, Universidad de las Ameritas-Puebla, 2001.
14. Senador de Siqueira, Thiago. "Bluetooth – Características, protocolos y funcionamiento". Instituto de computación, Universidad Estatal de Campiñas, 2006.
15. Fernández, Eduardo. "Wi-Fi: nuevos estándares en evolución", Centro de Difusión de Tecnologías ETSITUPM, Ceditec, 2007.
16. Blázquez del Toro, Luis Miguel. "¿Qué es RFID?". Sistemas de i∂dentificación Por Radiofrecuencia, Departamento de Ingeniería Telemática. Universidad Carlos III, Madrid, España, 2006.

17. AXCESS International Inc. “An Automated RFID Solution for Physical IT Asset Management and Protection”. 2006.

18. Innovision Research and Technology. “Turning the NFC promise into profitable, everyday applications”. Near Field Communication in the real world, 2006.

19. Ciguere, Eric. “Databases and MIDP, Part 1: Understanding the Record Management System”. [Consulta: Septiembre 2007], disponible en Web: http://developers.sun.com/mobility/midp/articles/database rms/.

20. Feeley, Peter. “Finding the Right Memory for Your Mobile Phone Design”. Application Design Manager, Mobile Memory Group, Micron Technology, 2005.

21. TitanicLinux.Net. “DaoGen Manual”. [Consulta: Septiembre 2007], disponible en Web: http://www.titaniclinux.net/cms/FC?link=daogenmanual.

22. Sun Developer Network (SDN). “Core J2EE Patterns - Data Access Object”. Sun Microsystem. [Consulta: Septiembre 2007], disponible en Web: http://java.sun.com/blueprints/corej2eepatterns/Patterns/ DataAccessObject.html.

23. Ortiz, Enrique. “Understanding the Web Services Subset API for Java ME”. Sun Microsystem. [Consulta: Agosto 2007], disponible en Web: http://developers.sun.com/mobility/midp/articles/webserv ices/.

CURRÍCULOS

Ricardo Andrés Fajardo. Recibirá el título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca, Colombia. Actualmente es miembro del grupo de interés de desarrollo de aplicaciones móviles e inalámbricas w@pcolombia, donde centra su investigación en el desarrollo de servicios ubicuos, aplicaciones para dispositivos móviles y la construcción de protocolos de servicios ubicuos para ambientes móviles.

Víctor Fabián Miramá. Recibirá el título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca, Colombia. Actualmente es miembro del grupo w@pcolombia, donde centra su investigación en el desarrollo de servicios ubicuos, aplicaciones para dispositivos móviles y la construcción de protocolos de servicios ubicuos para ambientes móviles.

Oscar Mauricio Caicedo. Ingeniero en Electrónica y Telecomunicaciones, Universidad del Cauca, Popayán, Colombia, 2001. Especialista en Redes y Servicios Telemáticos, Universidad del Cauca, Popayán, Colombia, 2002. Magíster en Ingeniería, Área Telemática, Universidad del Cauca, 2006. Coordinador del Grupo de Interés en Desarrollo de aplicaciones móviles e inalámbricas W@PColombia y miembro del Grupo de Ingeniería Telemática. Docente del Departamento de Telemática de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca.

Áreas de interés: Desarrollo de Aplicaciones y Servicios Telemáticos para Dispositivos Móviles, Servicios IP de nueva generación, Interoperabilidad de Aplicaciones y Redes Inalámbricas.

Javier Mesa Durango. Recibió el título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca, Colombia, en 2005. Actualmente cursa su segundo año de maestría en Ingeniería con Énfasis en Telemática en la Universidad del Cauca. Como miembro del Grupo de Ingeniería Telemática GIT y del semillero de investigación Grupo de Interés en Aplicaciones inalámbricas y móviles "W@pColombia", centra su investigación en el desarrollo de aplicaciones y servicios móviles ubicuos.

Francisco Orlando Martínez. Ingeniero en Electrónica y Telecomunicaciones, Universidad del Cauca, Popayán, Colombia, 2003. Candidato a Magíster en Ingeniería área Telemática, Universidad del Cauca. Coordinador del Grupo de Interés en Desarrollo de aplicaciones W@PColombia y miembro del Grupo de Ingeniería Telemática. Docente del Departamento de Telemática de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca. Áreas de interés: Desarrollo de Aplicaciones y Servicios Telemáticos para Dispositivos Móviles, Servicios IP de nueva generación y Redes Inalámbricas.